Universidade de Aveiro Departamento de Electrónica, Telecomunicações e Informática,
2010

Ricardo Vilaça Moreira

Infraestruturas para localização baseadas em redes sem fios
Wireless-based infrastructures for indoor localization systems

Universidade de Aveiro Departamento de Electrónica, Telecomunicações e Informática,
2010

Ricardo Vilaça Moreira

Infraestruturas para localização baseadas em redes sem fios
Wireless-based infrastructures for indoor localization systems

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Electronica e de Telecomunicações, realizada sob a orientação científica do Prof. Doutor Jose Alberto Fonseca e do Prof. Doutor Arnaldo Silva Rodrigues de Oliveira, Professores do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

the jury

presidente / president

Doutor Nuno Miguel Gonçalves Borges de Carvalho
Professor Associado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

vogais / examiners committee

Doutor José Alberto Gouveia Fonseca
Professor Associado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro (orientador)

Doutor José Carlos Meireles Monteiro Metrôlho
Professor Adjunto do Departamento de Engenharia Informática da Escola Superior de Tecnologia do Instituto Politécnico de Castelo Branco

Doutor Arnaldo Silva Rodrigues de Oliveira
Professor Associado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro (co-orientador)

acknowledgements

This dissertation comes as the result of some months of work and cooperation with different people and entities. Just to mention a few, I must leave a word of appreciation to my orientators, Dr. José Alberto Fonseca and Dr. Arnaldo Oliveira, to Carlos Amador, with whom many ideas and solutions were discussed, to Paulo Bartolomeu from Micro I/O for all the support and to the company for providing some of the devices that I needed to work with.

In a less academic way, I thank my family for their constant support throughout the last few years, specially the last one while I was working on this dissertation. Also, special thanks directed to all my good friends like Marek, Daniel (Chaves), Ricardo (Ciclista), Carlos (Vetras), Jorge (Condutor), Nuno (Paraquedista), Vitor (Sr. Vitor), all the international students that chose Aveiro for their mobility period this year and all the other people that helped, with me, to create the ESN section in Aveiro and with whom I spent a great amount of my "free" time. Last but, definitely, not the least, special thanks to the Polish "mafia" from Aveiro, in particular to Tatiana, not only for the great friendship and moral support but also for her constant preocupation and all her good advices that, surely, helped me in making this a better (and easier) dissertation.

Palavras-Chave

IEEE 802.11, IEEE 802.15.4, Wireless Sensor Networks, Localização, Gateway, Tag, TCP/IP

Resumo

A rápida evolução das tecnologias de redes sem fios, em particular na área das *Wireless Sensor Networks* (WSN), levou ao crescimento da necessidade de obter informações, sobre qualquer coisa, em qualquer lugar. As WSN têm sido amplamente utilizadas, por exemplo, em aplicações da área da domótica, em parte devido ao seu relativamente baixo custo e baixo consumo de energia. Estas características fazem com que este tipo de redes seja bastante importante no preenchimento de algumas lacunas ainda existentes noutras aplicações, como por exemplo, aplicações de localização de objectos.

Esta dissertação foca-se, então, na crescente importância e necessidade de localização de pessoas e objectos e apresenta uma das possíveis abordagens ao problema de localização em espaços fechados. Através da integração mútua de duas tecnologias *wireless* diferentes (IEEE 802.11 e 802.15.4) é possível tornar as comunicações wireless bastante mais eficientes em alguns campos de aplicação. Esta integração é conseguida com a criação de um *Gateway* com suporte para ambos os standards referidos e tradução quase transparente (para o utilizador) entre ambos. Assim, passa a ser possível a comunicação entre dispositivos mais baratos (802.15.4) e redes 802.11 baseadas em TCP/IP *(Transmission Control Protocol/Internet Protocol)*, abrindo algum espaço à criação de inúmeras aplicações, mais especificamente, aplicações relacionadas com localização, que representam, no fundo, o objectivo principal deste projecto.

A criação das infraestruturas necessárias e do respectivo *software* de controlo, ainda que esta solução possa ser aplicada em muitos outros campos, foi direccionada especificamente para cumprir com alguns dos requisitos do sistema de localização que venha eventualmente a ser criado. Assim, para além dos *Gateway*s, que fazem a tradução entre os dois standards já mencionados, este projecto consiste na criação de *Tag*s, pequenos dispositivos terminais de muito baixo custo e muito baixo consumo energético, que poderão ser acoplados aos objectos que necessitem de localização. Estes dispositivos comunicam com os *Gateway*s através de uma ligação wireless usando o standard IEEE 802.15.4 e, indirectamente, anunciam a sua localização através da disponibilização do valor da força do sinal na conexão. Estas informações são processadas e podem ser acedidas a partir de qualquer computador pessoal que esteja munido de um browser web e ligado à mesma rede.

Keywords IEEE 802.11, IEEE 802.15.4, Wireless Sensor Networks, Localization, Gateway Tag, TCP/IP

Abstract

The rapid evolution of Wireless technologies, specifically in the area of *Wireless Sensor Networks* (WSN), has led to a growing demand for the availability of information anywhere, about anything. WSN are being widely used, for example, in domotic applications, partly due to their relatively low cost and their low energy consumption nature. These characteristics make this kind of networks very useful to fill some gaps in other applications, e.g. localization-related applications.

This dissertation focuses, then, on the growing importance and demand for localization of people or objects and presents one of the possible approaches to the problem of indoor localization. By merging two different wireless technologies (IEEE 802.11 and IEEE 802.15.4), it is possible to make wireless communications more efficient for some applications. The merge is achieved by the creation of a *Gateway*, with support for both standards and near-transparent translation between them. This allows cheaper 802.15.4 devices to communicate with regular 802.11 TCP/IP *(Transmission Control Protocol/Internet Protocol)* based networks, giving space for inumerous applications, specifically for localization purposes, which is the main goal of this project.

The creation of the necessary hardware infrastructures and its respective control software, even though the solution may be used in a large range of applications, is more specifically directed to comply with some of the requirements of the localization system to be created, eventually. So, in addition to the *Gateway*s that make the translation between the two already referred standards, this project consists in the creation of *Tag*s, small low-power and low-cost end-devices that are to be attached to the objects in need of localization features. These devices communicate with the *Gateway*s by means of an 802.15.4 connection and indirectly advertise their location, by providing the signal strength of the connection. This data is computed and can be accessed in any personal computer with a web browser and a connection to the network.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Overview

This project is a branch of a larger project, **DHT-Mesh**. The parent project has the main objectives of addressing scalability problems in 802.11s networks and, by doing so, contribute to the development of WMN - *Wireless Mesh Networks*.

The 802.11s protocol is a Wireless LAN technology with growing interest and several improvements have been made to it recently, mainly concerning Media Access Control and Routing features.

Other mesh networking protocols exist, like the IEEE 802.15.4, which is already quite well developed. As it will be seen shortly, this project intends to make use of different mesh networking technologies for applications where 802.11 technology is unnecessary or unbearable.

1.1 Motivation

The rapid evolution of wireless communication technologies has given space and importance to the development of new devices and techniques which could fill the growing need for fast and reliable access to various kinds of information. In the same way, the possibility to locate people or objects has evolved from a mere curiosity issue to a necessity.

While some localization techniques are already very well developed (i.e. GPS, which works well outdoors), there are still some gaps, mainly concerning indoor localization of people or objects. The means to fill that gap have been widely available for some time (*IEEE 802.11, IEEE 802.15.4* and other wireless technologies), however, only a limited amount of research has been made in this area. There was still the need to integrate these technologies in the most efficient way, in order to get the most of them.

As it is known, the IEEE 802.11 standard is widely implemented in various electronic devices (laptops, PDA, cellphones...). This standard allows these devices to communicate between each other or with fixed Access Points, which makes it possible (with the assistance of some software) to track and locate the relative position or the proximity of the devices.

These devices have a relatively good energetic autonomy and, for a diversity of reasons, need to use many of the functionalities provided by the 802.11 standard.

A problem arises when it comes to the localization of devices or objects with lower energy resources, that can't handle the task of powering an 802.11 communication interface for long periods of time, or devices that simply don't need all the speed and robustness of this standard. For the efficient localization of such devices, a different standard may be used, such as the IEEE 802.15.4. This standard provides less resources than the previously mentioned one but, in exchange, has a much better energetic efficiency (i.e. less energy consumption) and is usually cheaper and easier to implement.

1.2 Objectives

This dissertation focuses on the creation of a system that, by combining the two wireless technologies mentioned before (IEEE 802.11 and IEEE 802.15.4), is able to provide connectivity between mobile and fixed devices at a reasonably low cost. The desired solution allows 802.15.4 devices to communicate through common 802.11 wireless networks as if they were 802.11 devices.

By achieving this objective, it becomes possible, for example, to view in a web browser the position of objects or people. The system must be able to provide the RSSI *Received Signal Strength Indicator* for signals sent by the mobile devices (*Tags*) to the fixed *Gateways*.

1.3 Document Structure

This dissertation is structured as follows:

- Chapter 2 - **Localization based on standard 2,4GHz RF technologies** - This chapter comprises an overview about Radio Frequency based technologies, namely the ones used for the purpose of indoor localization of devices, as well as definitions of some concepts that may be of assistance in the comprehension of this dissertation. Wireless technologies IEEE 802.11 and IEEE 802.15.4 are mentioned and briefly explained, along with information about RSSI, a very important concept, and others.

- Chapter 3 - **System Specification and Architecture** - In this chapter, the system architecture is described with limited detail, aiming to give an overview of the system and its mode of operation. Additionally, it includes some constructive remarks, focusing on and comparing the advantages and limitations of the system, along to the points that may be considered as limitations but fall outside the scope of this project.

- Chapter 4 - **Related Work** - Some notes on another project, with the same objective as this one, that followed a slightly different approach. Some characteristics from both projects are being combined in order to create an even more complete solution.

- Chapter 5 - **Retrieving and Delivering Localization Data** - Includes technical information and details about the mode of operation of the system, with the hardware and software used and created to implement the solution being explained to some extent.

  A more extensive guide on how to make use of the developed software is included in the appendix in the present dissertation along with instructions on how to use the development environment (*Microchip MPLAB IDE*) to create, modify and compile applications to run in the used hardware.

  In order to better illustrate the operation of the system, this chapter has been divided into some subsections that dissect the journey of a packet since it leaves the *Tag* until it is put available to the main server. Additionally, as the system is intended to provide only the connectivity support for RF based localization applications, there is a section concerning the availability of data and explaining how to access and use it.

- Chapter 6 - **System Validation** - This chapter focuses on validating or certifying the correct and desirable functioning of the system. The validation through demonstration is described here, with notes on how a specific demonstration works, how to prepare it, and how to get and analyze the results.

  Additionally, it includes the results of some tests, which can serve as guidelines for the configuration of some aspects of the system.

- Chapter 7 - **Conclusions** - As a finishing point to this dissertation, this last chapter is nothing more than a short debrief, with notes on the work already concluded and some final remarks regarding possible future improvements.

Chapter 2

Localization based on standard 2,4GHz RF technologies

2.1 Introduction

This chapter comprises a brief overview of the most common standard wireless technologies operating in the 2,4GHz ISM (*Industrial, Scientific and Medical*) frequency band, along with some other important concepts, related to this dissertation.

2.2 Fundamental Concepts

- **IEEE 802.11**

  The IEEE 802.11 is a very well-known standard for wireless communications. It is used in wireless applications with high-bandwidth requirements, making it the predominant standard in WLAN - *Wireless Local Area Network*s. Most laptop computers nowadays are equiped with an 802.11 interface and use it to connect to the Internet. This standard offers connectivity at relatively high speeds, at the cost of increased hardware complexity which, in turn, requires that the interfaces consume considerable amounts of energy.

- **IEEE 802.15.4**

  The IEEE 802.15.4 standard, when compared to 802.11, is much slower and less complex or complete. This standard offers low speed communications but compensates for it by requiring very low energy resources and being easier to use or implement. IEEE 802.15.4 is, then, mainly used for low power, low cost and low speed applications.

– **ZigBee**

ZigBee is a network protocol, created to standardize and uniformize the mode of operation of 802.15.4 Wireless Networks. It works over the PHY and MAC layers of the 802.15.4 standard and adds functionality to the 802.15.4 networks such as Routing Capabilities, improvements in Security and other general improvements, namely concerning the energetic efficiency and the robustness and reliability of the communications. The figure 2.1 represents the Zigbee protocol stack using IEEE 802.15.4 specifications, specifically its Medium Access (MAC) and Physical (PHY) layers.

Figure 2.1: ZigBee Protocol Stack

The main characteristics of the ZigBee protocol stack are [2], among others:

* Data rates of approximately 250 kbps
* Best suited for low duty-cycle applications
* CSMA-CA
* Low latency
* Low cost
* Low power, hence, long battery life
* Multiple network topologies: star, peer-to-peer, mesh
* Optional guaranteed time slot for low latency applications
* Fully hand-shaked protocol for transfer reliability

A ZigBee network may assume various types of configurations. The three main types are the already referred *star*, *peer-to-peer* and *mesh* configurations.

Standard	Speed	Power Reqs.	Advantages	Applications
IEEE 802.11 (b/g/n)	High	High	High Speed, Range	Large data transfers
IEEE802.15.4	Low	Low	Low Power, Cheap	WSN, WPAN, Small transfers

Table 2.1: Technology Comparison

- **RSSI - *Received Signal Strength Indicator***

  Among other alternatives, checking the RSSI is one of the best ways to learn about the strength of a wireless connection. The value of RSSI is based on the measurement of the energy of a received radio signal. This value is inversely proportional to the physical distance separating both ends of the radio link, hence it can be used to estimate with some precision the proximity of a radio signal source.[3]

  One of the disadvantages of this method lies on the fact that an RF transceiver may receive *interference energy* from sources operating in the same channel, other than the source that is actually communicating with it. This may lead to incorrect measurements and should be taken into account when considering using RSSI readings for high-precision or critical applications.

  There are other indicators that could be used to check the quality of a wireless signal. One of the most complete ones is **LQI** - *Link Quality Indicator*, which is the result of a combination of RSSI with other values like the percentage of packets successfully received, for example. This indicator is useful for routing protocols, specially in mesh networks, because it constantly gives an indication of the link quality (that doesn't depend only on the signal strenght but also on other factors like the percentage of occupation of the nodes) allowing the routing protocol to always choose the routes according to the probability of the message being delivered successfully.

- **TCP/IP** [4]

  The ICS - Internet Control Suite (sometimes referred to as TCP/IP, because TCP and IP were the first two protocols to be defined in this suite) is a set of protocols used for facilitating and standardizing communications between different hosts in a network.

  The TCP is the responsible for making the interface between a user or application process and lower level protocols (such as IP). This interface consists of a set of functions that processes can call whenever necessary in order to perform some specific tasks. These tasks include, for example, opening and closing connections and sending or receiving data through already established connections. In order to provide a quality service, while working on top of less reliable communication layers, TCP requires facilities in the following areas:

  - Basic Data Transfer
  - Reliability
  - Flow Control
  - Multiplexing
  - Connections

– Precedence and Security

To assist in some of these areas (specially *Multiplexing* and *Connections*), the concept of *socket* was introduced.

– **Socket**

In a single host, the applications communicate between themselves, internally. In a network with multiple hosts, a similar situation occurs, being that applications running in different hosts can communicate with each other. This is still considered as IPC - Inter Process Communication.

As processes may need to distinguish between various communication streams (internally or with other hosts in the network), each communicating process in a host can be uniquely defined by a concatenation of the hostname or address with a port number, assigned to the process by the host. This allows for the creation of *Sockets*, that can be thought of as a virtual "plug" which can be used to connect to a host. A connection can, thus, be fully specified by a pair of sockets (a socket at each end of the connection). The same socket may, additionally, connect to multiple sockets in different hosts simultaneously.

– **Berkeley Sockets**

The **Berkeley Sockets** API is a collection of primitives that allow simple usage and manipulation of socket-related tasks like the creation of TCP sockets. It is the most standard abstraction mechanism for dealing with network sockets. It was developed at the University of California, Berkeley, to be used in UNIX-based systems (namely Berkeley's own operating system, BSD). Nowadays it is widely used in all UNIX-based operating systems and has been ported to non-UNIX operating systems and various platforms.

Some of the primitives comprised in the Berkeley Sockets API are:

* **socket()** - creates a new socket
* **bind()** - associates a socket with a port number and an IP address (tipically used on the server-side of the connection)
* **listen()** - also used on the server-side, causes a socket to enter the listening mode (waiting for conenctions from other sockets)
* **connect()** - used on the client-side, assigns a port number to the socket and attempts to connect to a specific destination socket
* **accept()** - cause the listening socket to accept a connection from another socket
* **send()** and **recv()** - transfer data between the connected hosts

The above concepts are nothing more than a way to introduce some of the terms and tools that are used throughout this dissertation.

2.3 State of the Art

Lately, a great amount of RF based localization protocols have been developed. Some of them were created only with investigational purposes, others with military intentions and many are available to the general population and represent an important part of people's everyday life.

It is known and obvious that wireless communications are greatly influenced by several conditionants related to the surrounding environment, namely the amount of wireless devices communicating in the same frequencies, which can cause collisions and induce delays or breaks in communication. For this reason, a lot of solutions have been researched, in a quest to find the best protocol for each desired area of application.

Regarding wireless based localization systems, the most widely known example is the GPS, that relies on satellite triangulation and a good knowledge of the planet to provide its users with information about their location GPS is, however, a solution that only works well outdoors; for indoor localization systems other methods must be used (because the satellites are not usually easily accessible from within closed spaces).

Some of the methods already used to assist in the implementation of wireless-based localization systems are described in the following lines:

- RSSI - The Received Signal Strength Indicator is a measurement of the energy in a radio signal. According to the energy of the received radio signal, a receiving device can calculate with some precision the distance to the source of the same radio signal. From general knowledge and simple logic thinking, it is known that the signal strength decreases when the distance to the source increases. This method may not be the most precise one but is precise enough to give an indication about the location of devices. The method becomes more precise and reliable if the physical and environmental characteristics of the place where the localization must be performed are well known, since it allows for the calibration of the localization devices.

- Time Difference of Arrival - As the radio signal waves travel at approximately constant speeds (assuming a more or less uniform transmission media), it is possible to determine the location of a radio signal source by measuring the time that a signal needs to reach a receiver.

- Triangulation - Following the same principle of the GPS technology but relying in local wireless Access Points sitting on a known location instead of satellites, this method is effective but its implementation is quite complicated and expensive.

The methods mentioned above are already used, however, the resources needed to use them are, sometimes, quite unaffordable (not necessarily for financial reasons). Localization

of, for example, a laptop computer with a standard wireless card may be performed using any of the mentioned methods.

This dissertation presents a solution based on the RSSI method for indoor localization of objects, however, it is not based solely in common wireless cards as in the example mentioned. Instead, it takes advantage of other technologies to assist in the creation of a more efficient solution.

Chapter 3

System Specification and Architecture

3.1 Executive Summary

Aiming to find a low-cost and high-efficiency solution that could provide information about the location of people or objects, various hardware modules were considered, with the final option falling over *Microchip* devices. This choice was made due to, not only the fact that *Microchip* had the necessary hardware for the desired implementation, but also because of previously acquired experience working with similar devices. The technologies chosen to approach this challenge were standard wireless technologies, namely IEEE 802.11 and IEEE 802.15.4. The first, for its ease of implementation and compliance with standard wireless networks; the second because of its low-power and low-cost nature.

3.2 Requirements and Conceptual Architecture

The main function of the *Gateway* is to, transparently, allow the exchange of information between IEEE802.11 and IEEE802.15.4 networks.

Figure 3.1 represents a simplified view of the system with all its core components and some additional blocks.

Figure 3.1: Simplified Operation Diagram

The 802.15.4 standard has limited capabilities, namely concerning its speed (approximately 250kbps, maximum). However, as the system will be mainly used for localization purposes, specially for the indoor localization of people or objects, there is no need for a very high-speed system (it would only increase its cost and complexity). The location of people or objects inside a building usually changes at a relatively low speed (when compared to the usual speeds of wireless communications and data processing) so, if the details

about the location of objects are only updated once every second, they can be considered relatively accurate. Furthermore, the system is intended to provide data for localization oriented only to the compartment (i.e. it is not necessary to know the exact location of something inside a compartment). The flexibility associated with the loose precision requirements may allow for a less comples and less expensive system overall, along with a foreseen lower network occupation percentage (*per* device).

3.2.1 Simplified Operation

The core of the solution consists of two kinds of devices:

- complex devices, which will be referred to as *Gateways* from now on, which transparently merge the two technologies (IEEE 802.11 and IEEE 802.15.4).
- simpler devices, with lower resources and lower energy consumption, referred to as *Tags*, which communicate using IEEE802.15.4 technology, exclusively.

In addition, a web server, a standard 802.11 wireless Access Point and some other "blocks" are represented, to better illustrate the mode of operation of the system in a simpler way. These "blocks" are necessary for the intended operation of the system but are only mere accessories, not considered as integrants of the core of the system.

The *Tags* are small portable devices that, periodically, send a broadcast message to the air. This message is received by the *Gateway*s that are in range and contains, among others, information about the *Tag* that sent it and the strength of the received signal. These informations are useful to calculate the approximate distance between the transmitting *Tag* and the receiving *Gateway*(s).

After receiving a message, the *Gateway*(s) send it through their IEEE 802.11 connection to a web server. The web server is responsible for selecting the valid data (i.e. in cases where more than one *Gateway* receive the same message and forward it to the web server, only the one with higher signal strength indicator shall be selected) and computing it according to the needs of the localization application. For the purpose of this dissertation (indoor, compartment-oriented, localization of objects) and knowing the limitations of the (around) 2,4GHz frequency band regarding its ability to pass through walls, in general, it is safe to assume in most cases that the *Gateway*s that receive the message with the higher associated RSSI is the one that should be listened to (because probably it is the one closest to the emitting *Tag*).

The data can then be displayed in a visually attractive manner (in a common web browser or other purpose-built applications), allowing the users to easily collect informations about the location of the *Tags*. This information may then be put available to users in the local network or anywhere in the world wide web.

Example of application...

An example of application is shown in figure 3.2. This figure represents a simple example of a floorplan (two compartments, one of which is divided into two areas) where the

system has been installed. There are three *Gateway*s and four *Tag*s. Each *Tag* periodically broadcasts a message to be listened by any *Gateway* in range. The *Gateway*s that receive messages, forward them to the Access Point which, in turn, makes them available to be consulted on any computer (intranet or internet) with a web browser.

Figure 3.2: Example of Application

The information is displayed on the computers as shown in figure 3.3 and it serves the informative purpose, even though it can be processed in different ways and be presented with a more appealing visual aspect. This information is nothing more than a collection of the last readings of each *Gateway* and its association with the location of each *Gateway* and its relative distance from the transmitting *Tag*s.

Figure 3.3: Localization Data displayed on a web browser

3.3 Contributions for the Work in Progress

There are some details, regarding this implementation, that could pose us problems in some situations. For example, the fact that the wireless Signal Strength between two devices depends on factors other than the distance between them (i.e. obstacles, quantity of radio-frequency devices in the area, etc...), can make RSSI-based localization a little unreliable. This situation can be partially solved by "calibrating" the devices, adjusting them according to the conditions of the place where they are to be installed.

Another issue that may be foreseen in this implementation is related to the *non-real-time* characteristics of the system. This, however, given the flexibility and the *non-critical* nature of the predicted applications can be considered a minor issue (for indoor, compartment-oriented, localization purposes, resolutions in the order of one or even more *seconds* can be acceptable).

Apart from the mentioned issues, there are potential problems that could arise in certain situations. A crowded environment (with a large amount of wireless devices connected to the same infrastructures) could lead to an increase in the packet loss percentage due to collisions. There are several possible ways to address this problem, however, this project doesn't intend to focus on them.

The project has well defined goals that pass by creating the necessary base infrastructures for communication; details like the mentioned one fall out of the scope of the current project or may be eventually incorporated in future work.

Chapter 4

Related Work

Inserted in the same project (*DHT-Mesh*) and to serve a purpose very similar to the system described in this dissertation, a different solution is being developed. The referred solution is a work in progress (but already functional) and is more directed to the localization of devices that don't have a lot of energetic limitations, as it uses different, more powerful, infrastructures.

It consists of a wireless Access Point (AP) to which is physically connected a Zigbee (XBee) transceiver. The AP runs an *open-source* Linux-based operating system, with the necessary drivers to interface with the XBee module. This block has the same purpose of the *Gateway*s developed for the current dissertation, with the advantage of being able to create standard 802.11 networks or connecting to Ethernet IP networks.

As the *Gateway*s must connect to a wireless Access Point in order to forward the information received from the *Tag*s, the idea of integrating the two projects is not only feasible but also very interesting. This led to a collaboration, namely in the development of a web-interface, hosted in the Access Point running an http daemon (web server), which allows for displaying informations in a web browser without the need of a separate web server.

Basically, this alternative solution merges the Access Point with the webserver, resulting in a less expensive solution. The system shown in figure 3.1 becomes the system in figure 4.1.

Figure 4.1: Simplified Operation Diagram - AP with embedded web server

Additionally, as the customized Access Point being used has been equiped (as referred above) with a XBee transceiver, it allows for XBee 802.15.4 devices to connect to it and provide localization information in a similar fashion to the one presented in this dissertation (but with different modules).

More details related to this different implementation may be found in an on-going Msc dissertation entitled "Wi-Fi/ZigBee AP to support localization based on wireless networks"

Original Title, in Portuguese: "AP Wi-Fi/ZigBee para Suporte à Localização Baseada em Redes Sem Fios"[5].

Chapter 5

Retrieving and Delivering Localization Data

5.1 Outline

In order to provide connectivity between different technologies, there was the need to search the market and choose the more adequate between various communication modules, as well as a microcontroller unit, to control the modules.

This chapter includes some technical information about the chosen communication modules. At first, there is a brief presentation about each module and its main characteristics, specially the ones with a higher importance for the purpose of this localization system. Further sections explain the mode of operation of each communication module, along with information on the operation of the two core elements of the system (i.e. *Tag*s and *Gateway*s) and on how the modules integrate in these elements.

For a better understanding of the operation of the system, this chapter also details the various procedures that a message has to go through since it leaves a *Tag* until it passes the *Gateway* and reaches its final destination (i.e. a web-server). The additional changes that the message may suffer after reaching the web server depend on the desired application and fall out of the scope of this dissertation.

5.2 Infrastructures

The chosen microcontroller was a PIC from *Microchip* (dsPIC33FJ256MC710). Additionally, two wireless modules were chosen; *Microchip MRF24J40* and *ZeroG/Microchip ZG2100MC*, for 802.15.4 and 802.11 connectivity, respectively. The choice of *Microchip* modules to implement connectivity has been made due to the ease of use, high compatibility and availability of information from the manufacturer.

5.2.1 uMRF "development" board

To provide support and connectivity for the microcontroller during the prototyping stages of the project, MicroI/O created a development board called μMRF.

The μMRF board includes, apart from the microcontroller, the following peripherals:

- Temperature Sensor
- Accelerometer
- USB connection
- Push Buttons
- LEDs

The board also includes an 802.15.4 transceiver (*Microchip* MRF24J40) that will be used in this project.

Additionally, the μMRF board includes two special connectors that add expansion capabilities to it. *Mezzanine* boards can be connected to the μMRF board very easily.

5.2.2 *Tags*

Architecture

The *Microchip* MRF24J40 module (fig. 5.1) is an RF transceiver that operates in the 2,4GHz frequency band. It connects to the microcontroller by means of a 4-wire SPI (Serial Peripheral Interface) connection, as well as through *interrupt*, *reset* and *wake* pins. This module "*is compliant with the 802.15.4-2003 standard, which specifies the physical (PHY) and Media Access Control (MAC) functions that form the basis of a wireless network device.*"[1]

Figure 5.1: Microchip MRF24J40 - 802.15.4 Transceiver

The MRF24J40 provides hardware support for, among others:

- **Energy Detection** - allows the calculation of RSSI values.
- **CCA** - *Clear Channel Assessment* - this functionality allows the device to check the occupation of the transmission media (air) before attempting to transmit a packet. This helps avoiding collisions.
- **CSMA-CA algorithm** - *Carrier Sense Multiple Access - Collision Avoidance* - this algorithm also prevents collisions from occuring.

- **Automatic Acknowledge** (and automatic packet retransmission, for a limited number of times, in case of failure).

- **Energy Saving** - the module supports various energy saving related features (sleep mode, wake-on-interrupt and others)

This module includes a limited amount of memory, implemented as static RAM, which can be accessed and written to via the SPI port on the module. The memory is functionally divided into two "sections", control registers and data buffers (FIFOs), as shown in figure 5.2.

Figure 5.2: MRF24J40 - Memory Organization [1]

The Control Registers are used for status checking, control and addressing purposes. The FIFOs are mere temporary buffers, used for data transmission or reception by the MRF24J40, as well as to support some security features. There are four FIFOs to handle transmission (*TxFIFO*), each with 128 bytes, and one 144-byte reception FIFO (*RxFIFO*). The *RxFIFO* can hold any kind of packets, as long as they are compliant with the 802.15.4 standard. The *TxFIFO*s, in turn, have different functionalities, being that each buffer is used to transmit a different kind of packet.

Throughout this dissertation, only "Normal" packets are mentioned (as other kinds of packets are not necessary for this thesis purpose) and the "TX Normal FIFO" will be referred to as "TxFIFO" from this point on.

The *TX Beacon FIFO* can be used for the transmission of *Beacons* (broadcast frames that assist in the synchronization between devices connected to the same network). This, in turn, provides support for GTS (*Guaranteed Time Slots*), a feature that can prove itself valuable for real-time applications and that uses the *TX GTS1 FIFO* and *TX GTS2 FIFO*.

Whenever a message is written to the TxFIFO, the MAC automatically formats it for transmission according to the IEEE 802.15.4 standard. The CSMA-CA functionality is implemented in hardware, as well as some of the already mentioned security features.

The μMRF board incorporates this module, connecting to it through the microcontroller's SPI1 interface and INT1 interrupt line.

Figure 5.3: uMRF development board - battery powered

5.2.3 *Gateway*

Architecture

The *Gateway* is the main point of interest of this thesis. It is the responsible for the merging of two different technologies and establishing connectivity between devices with either one of them. To achieve this goal, it must be provided with the necessary physical interfaces.

In a nutshell, the gateway is composed by three main blocks:

- Microcontroller (Microchip dsPIC33FJ256MC710)
- IEEE 802.15.4 Transceiver (Microchip MRF24J40)
- IEEE 802.11 Module (Microchip/ZeroGWireless ZG2100)

Figure 5.4: Microchip/ZeroG ZG2100 - 802.11 Transceiver

It is easily noticeable that a *Gateway* is merely a *Tag* with an extra interface (802.11). Obviously, the control software must behave differently than the software prepared for the *Tags*. So, in order to build a *Gateway*, we can use the *mezzanine* connectors on the uMRF board to attach a ZG2100 module (fig. 5.4) and provide it with power and communication through the SPI2 interface of the microcontroller.

Additionally, the ZG2100 module is connected to the one of the microcontroller's interrupt lines (INT2).

5.3 Retrieving RSSI

5.3.1 Tag Operation

In the context of this application (localization), the *Tags* only really need to transmit (and not receive) packets periodically, to inform the *Gateway* of their whereabouts. Given this assumption, this section intends to focus more on the details related to the transmission of packets than related to the reception.

Each time that the microcontroller needs to send a message to the transmission media (air), it creates an array of bytes with the contents of the message, as shown in table 5.1. The array is then sent to the TxFIFO of the MRF24J40 module, through its SPI connection.

Field	Header Length	Frame Length	Frame Control	Sequence Number	DestID	SourceID	Payload
Size (bytes)	1	1	2	1	2	2	8

Table 5.1: 802.15.4 custom Tx Data frame format

The format and fields of the transmission frame, represented in table 5.1, has been adapted to fit the reality of this project; still, it maintains some compatibility with the ZigBee specification because it may be useful in the future. The decision to opt for almost-RAW packets instead of using a protocol stack (like the MiWi [6] or ZigBee stacks, both provided by *Microchip* for implementation in the used microcontroller) has been made mostly because, for the desired usage, there was no need for a very complex and heavy stack with routing protocols and other complex features. Transmitting custom packets without the need for a stack makes implementation, customization and usage a lot simpler. Furthermore, by avoiding even the most common stack operations, a lot of processing time and energy can be saved, along with other of the microcontroller and transceiver resources.

This custom implementation, however, even trying to maintain some compatibilty with the ZigBee specification, does not need to use all the fields in the frame in the same way as an actual ZigBee implementation; specifically, the fields that concern device identification, are handled differently. Details will be explained shortly, together with the reception frame details.

There are three types of packets/frames that can be transmitted by this device: Data, Beacon and Command frames. In this implementation, only Data frames will be transmitted by the *Tags*.

The fields in a Data frame are, then, as follows:

- **Header Length** - this field is pretty much self-explanatory. It represents the size of the Header (which, in turn, is composed by the Frame Control, Sequence Number and Addressing Fields)
- **Frame Length** - the Frame Length is the sum of the size of the Header with the Payload (Data)
- **Header**:
  - **Frame Control** - not used, but implemented to maintain compatibility with the ZigBee specification
  - **Sequence Number** - incremented for each new message, useful for the receiving end of the messages (to avoid receiving repeated messages and to detect lost messages).

  - **Addressing Fields:**
    * **DestID** - identification of the destination (receiver). The identification conventions will be explained in more detail, as they differ from regular ZigBee implementations.
    * **SourceID** - identification of the source (sender)

- **Payload** - space for generic raw data. Not used and not necessary in this implementation.

As soon as the module detects data in the *TxFIFO*, the MAC layer of the transceiver processes the message and appends a 2-byte field to the end of the packet. This field, the FCS - *Frame Check Sequence*, is composed by checksum characters, which may be used for error detection and correction purposes.

The packet is then passed to the PHY, which appends some control and synchronization fields to the beginning it and attempts to transmit it, right after checking the availability and occupation of the transmission media (making use of the CCA and CSMA-CA embedded capabilities already mentioned).

5.3.2 Gateway Operation

As previously explicited, the *Gateway* must behave like a buffer, that is, it must forward to the 802.11 interface all the information received from the 802.15.4 interface, and vice-versa when necessary. In this case, and to fulfill some of the requisites of the desired application (localization), the *Gateway* performs some adjustments to the information before forwarding it. Also, more attention will be given to the communication in the direction of 802.15.4 $\rightarrow$ 802.11 than in the opposite direction.

Figure 5.5: *Tag* $\triangleright$ *Gateway*

This section is divided into four different parts, for organization purposes and to allow for a better understanding of each of them and how they are related.

Connecting to an 802.11 Network

The connection to an IEEE 802.11 Wireless Network is automatic, as long as the *Gateway* has been configured with the correct parameters to connect to the desired network. Because the uMRF boards lack an EEPROM or some other kind of non-volatile memory, the configurations must be hard-coded (i.e. configuration options must be written in the application source code, before compiling and uploading it to the uMRF board).

By default, the *Gateway*s are configured to connect to any **open** (without authentication) wireless network with the SSID: "DHTAP". The Access Point must be running a DHCP Server and provide the *Gateway*s with an valid IP address, otherwise, the *Gateway*s

may assume their default IP address (172.20.30.44). In case there are more than one *Gateway*, their default IP addresses have to be configured in the source code, to avoid having different devices in the same network sharing a conflicting IP Address.

Authentication and the most common security features from the 802.11 standard are supported by the *Gateway*s but are disabled by default. In order to activate and configure them, the source code of the application running on the *Gateway*s must also be changed as desired.

Receiving packets from the *Tags*

The 802.15.4 messages sent by the *Tags* must be received by *Gateways*. The MRF24J40 module implemented in the *Gateway* serves this function as desired.

The module supports three reception modes:

- **Normal** - accepts only packets with a good CRC that also completely satisfy the requirements of the IEEE 802.15.4 specification.
- **Error** - accepts any packets with good or bad CRC.
- **Promiscuous** - accepts any packets with good CRC, even if they don't comply with the IEEE 802.15.4 specification.

As mentioned before, the messages sent by the *Tags* don't exactly follow the standard. To allow these messages to be accepted by the receiving end (*Gateway*, in this situation), its Reception Mode must be set to "Promiscuous". This can be achieved by setting the RXMCR register accordingly[1].

When a packet with good CRC is received by the MRF24J40 module, it is written to its RxFIFO and the microcontroller is signalled with a "Receive Interrupt" interrupt request. After receiving the interrupt request, the microcontroller must disable the interrupts as well as disable the reception of packets in the MRF24J40. This last step is necessary because, as the first byte of the RxFIFO is accessed for the first time, the wireless module assumes that all the data has been read and starts accepting other incoming packets, which may overwrite the (still) unread data.

The frame format of a received message is presented in table 5.2.

Field	Frame Length	Sequence Number	DestID	SourceID	Data	FCS	LQI	RSSI
Size (bytes)	1	1	2	2	8	2	1	1

Table 5.2: 802.15.4 custom Rx frame format

It is noticeable that the frame differs from the frame sent by the *Tag*. In fact, the last two fields (**RSSI** and **LQI**) are added by the receiving PHY, according to the quality of the radio link and signal power when receiving the message.

After the message has been received and read from the RxFIFO, it is written to a RAM buffer, which is made available for the necessary processing by the microcontroller.

Finally, the interrupts and reception ability of the MRF24J40 are re-enabled, and the system proceeds with its normal operation.

Delivering Localization Data

After the packet has been received by the MRF24J40 module, decoded and transferred to the microcontroller RAM, it should be forwarded through the 802.11 interface to the desired destination (main server).

Figure 5.6: *Tag* ▷ *Gateway* ▷ *Access Point*

As opposed to the 802.15.4 communication (for the current implementation), the communication over an 802.11 network must be kept in the most standardized level possible, in order to easily interface with regular 802.11 based networks and with the devices connected to it. To facilitate this task, *Microchip* provides a TCP/IP Stack to be used with its microcontrollers which revealed to be very adequate to achieve the desired objective.

The provided TCP/IP Stack has its own implementation of the *Berkeley Sockets API*. This API provides functions for the creation of TCP sockets, some of which have already been mentioned in this document, in section 2.2.

Using the referred API, it was possible to create a function (called "RF2TCPBridge") to perform tasks related to the frame formatting and sending of the packet to the 802.11 PHY. This function is, basically, an FSM (*Finite State Machine*) that verifies if there is already an open TCP socket, connected to the main server. In case that there is no connection, a socket is created. After the connection to the server has been established, the function checks if there is data to be sent and sends it through the TCP socket, making it available at a socket in the destination (main server). A simplified diagram of the referred FSM is shown in figure 5.7

Figure 5.7: Communication through a TCP Socket - FSM (client-side)

Data encoding...

The data received from the 802.15.4 interface is encoded in the data/payload field of a regular 802.11 Data Frame. As the 802.11 frames automatically include informations about the source and destination of the packet (among others), this data doesn't have to

be pushed into the data/payload field. The generic 802.11 frame format is shown in table 5.3

Field	Frame Control	Duration ID	Addr1	Addr2	Addr3	Seq.Ctl	Addr4	Frame body	FCS
Size (bytes)	2	2	6	6	6	2	6	0-2312	4

Table 5.3: 802.11 frame format

Most of the fields in the frame are automatically handled or created by the stack functions or by the lower layers of the 802.11 standard. For the purpose of this application, only the Frame Body field (which is intended to contain custom data) is important. This field will, then, be used to hold the data sent by the *Tag*s to the *Gateway* so that the interpretation of the frame on the server-side can be simpler and follow the same formatting rules as the custom 802.15.4 frames previously mentioned.

Technically, the "Frame Body" field will contain the contents of the 802.15.4 Rx Frame5.2, except for the "Frame Length" and "LQI" fields, as they are not necessary for the desired purpose.

5.4 Availability of Information/Data

As long as there is a connection to a wireless network, the *Gateway* tries to establish a connection with the server (through a TCP socket, as mentioned before), which will be used to provide the web server with data that can be computed and displayed to the *end-user*. As previously shown, the relevant data assumes different formats since it leaves the *Tag* until it reaches the web server, its final destination. To make it easier to "follow"' the data, the format transformations are clearly shown in figure 5.8. The data is provided using the format shown in table 5.3 and the server is responsible for computing and presenting it in the way that better suits the application. The server must create a TCP socket and accept connections from clients (which run in the *Gateways*) in order to be able to receive the necessary data.

Figure 5.8: Frame Evolution

Chapter 6

System Validation

One of the best ways to prove the correct functioning of a system is through demonstration. This chapter attempts to recreate a simple demonstration, allowing some functionalities of the system to be tested. Special attention is given to the performance of the gateway, by evaluation of its capacity to receive, process and forward messages.

6.1 Demonstration

The system to be demonstrated is composed by the already described modules. In this demonstration, additional resources are used to show the correct functioning of the system, namely a Wireless Access Point with a built-in web-server running a CGI web-interface and a computer.

6.1.1 *Tags ▷ Gateways*

Two *Tag*s and two *Gateway*s are used in the demonstration and both *Gateway*s are connected via 802.11 to the Access Point, as shown in figure 6.1.

Figure 6.1: Demonstration diagram - part 1

The arrows represent the flow of data, from each *Tag* to both *Gateway*s. The data is sent by the 802.15.4 *Tag*s at fixed periodic intervals or, optionally, at any instant by pressing a push-button in the *Tag*.

6.1.2 *Gateways ▷ Access Point*

The data is then forwarded from the *Gateway*s to the Access Point (via 802.11 connection) (see fig. 6.2) and provided to the embedded web-server, which makes it available to be accessed by a web browser running in the laptop. The machine running the web server

is also responsible for filtering repeated or invalid information. As the *Tag*s broadcast their messages, it is possible that more than one *Gateway* receive the same message. In this case, the web server receives the same message from various different sources and must be able to select the appropriate one, according to the strongest reported signal strength in the message from each source.

For some localization applications, specially if they involve triangulation (see chapter 2.3), "repeated" messages may be accepted.

Figure 6.2: Demonstration diagram - part 2

All the configuration parameters are hard-coded in the applications running on the *Tag*s and *Gateway*s. For this demonstration to work without the need to configure anything, the Access Point must be configured with the SSID: "DHTAP" and all security features must be disabled.

6.1.3 *Access Point/Web Server ▷ Personal Computer*

To access the data, a web browser must connect to the AP using its IP Address and a chosen port (default: 8888). These parameters may be chosen in the AP.

The web interface displays various informations, being the most relevant one the RSSI that each *Tag* last reported to its nearest *Gateway*.

6.2 Measurements

Conceived to be integrated in embedded systems and low-power devices, the communication modules used in this project have, obviously, limited capabilities. One of the most important characteristics that will be tested, is the speed at which they are capable of sending and receiving information.

Various tests were performed, in order to evaluate the capacity of such modules.

NOTE: All the tests were performed in an uncontrolled, yet aggressive, environment (regarding interferences from other wireless devices). The packets involved in the 802.15.4 part of the tests have exactly 1-byte each, unlike the larger packets used for localization purposes. This makes it easier to calculate the transmission speeds in Bytes/sec. The "data"' field of the 802.11 packets is 16-byte sized but, obviously, there is a significant amount of protocolar overhead that has to be taken into account.

- **802.15.4 Data Exchange Rate** (*Tag ▷ Gateway*)

  The *Tags* are programmed to send a custom packet everytime that one of the microcontroller's timer (TMR9) counts to a pre-configured value. The timer is incremented

with each clock cycle (in this application, prescaled clock, with a prescaling factor of 1:256).

The internal clock frequency is approximately 10MHz. The downscaling of the clock is necessary because the MRF24J40 modules are not able to transmit data at such high speeds (the maximum theoretical speed for 802.15.4 data transfers is around 250kbps). After downscaling, the timer is ruled by an input clock with a frequency of:

$$\text{T9CLK} = \frac{10MHz}{256} \approx 39.062\text{KHz}$$

To choose the *Tag*'s transmission rate, the PR9 register needs to be configured. This register must hold a numeric value which corresponds to the desired period of the Timer9 or, from a simpler point of view, the number of Timer9 input clock cycles until an interruption is triggered. Note that a packet is transmitted each time that Timer9 generates an interrupt request.

By testing it with different values and checking, for each value, the percentage of packets that were succesfully received (by the *Gateway*) and the percentage of packet loss, it is possible to find an acceptable value to set the transmission rate of the 802.15.4 *Tags* to.

If the running application didn't carry some overhead (related to other tasks), the transmission rate achieved (or attempted) by setting the PR9 register could be as high as 39.062KHz (when setting the value of PR9 to 1, the lowest value). At this frequency, and having the *Tag* and the *Gateway* separated by approximatelly one meter, the *Gateway* is not able to receive and process all the data sent by the *Tag*.

PR9	≈ TxFreq.(Hz)	Packets Tx	Packets Rx	P.Loss (%)
500	78.13	1000	366	63.4
700	52.08	1000	548	45.2
750	39.06	1000	730	27.0
1000	31.25	1000	907	9.30
1250	19.53	1000	999	0.10
2000	7.81	1000	1000	0.00
10000	3.91	1000	1000	0.00

Table 6.1: 802.15.4 Packet Loss

As can be seen in table 6.1, the packet loss stops to be significative only when PR9 is set to values around or above 1250, which is the equivalent to a transmission frequency

Figure 6.3: TCP/IP Builder - socket created in local interface (172.20.30.136)

of approx. 19,53Hz. The responsible for this situation is the overhead caused, mainly, by the "heavy"' application running in the *Gateway*. As, for localization purposes, a time resolution of one second is very acceptable, the value of 2000 has been chosen for the PR9 register. The transmission frequency is, then, set to approximately 7.81 Hz which is higher than necessary for the purpose of this application and, at the same time, low enough to avoid having to deal very often with colision problems in very populated wireless environments (the less packets in the air, the lower the chance of collisions occuring).

- **802.11 Maximum Transmission Rate** (*Gateway*)

  As the 802.11 standard, by definition, supports much higher speeds than the 802.15.4 standard, it could be assumed that the *Gateway* would be capable of forwarding all the packets through the 802.11 PHY at the same rate as they were being received by the 802.15.4 PHY. Even knowing that the ZG2100 module can only reach speeds up to 2Mbits/sec (256kBytes/sec), this is still about 128 times faster than the maximum transmission rate of the 802.15.4 devices. However, for documentation purposes and to take into account all the overhead and the interferences from other wireless devices, some simple measurements were taken.

  The test performed consists on simple packet transfers through the 802.11 interface, between the *Gateway* and a listening TCP Socket server (running in a personal computer). By testing the transmission of data at different speeds and analyzing the packet loss percentage for each speed, it is possible to find out the maximum speed of transmission for which the packet loss is minimum.

  The listening server runs a freeware program called *TCP/IP Builder* , which functions as a TCP socket server in the desired port (54321, in this case) and outputs to the screen (and, optionally, to a log file) all the information received through the created socket.

  Figure 6.3is a screenshot of the TCP socket server application used to test the proper reception of data. The local computer (server) IP is 172.20.30.136 and the socket is open in port 54321.

  The *Gateway* (IP - 172.20.30.132) connects to the socket server as a client and, periodically, tries to send a 16-byte predefined packet.

  The predefined packet has the following aspect:

  01:23:45:67:89:AB:CD:EF:EE:DD:CC:BB:AA:99:88:77

  To perform the test, Timer9 was set with a prescaling factor of 1:256 and various values of PR9 were tested; As explained in a previous section in this document, the

Figure 6.4: TCP/IP Builder - some of the received packets

PR9 is a register that controls the Counter Mode of Timer9. Its value represents the number of input clock cycles until a timer interruption is generated.

PR9	≈ TxFreq.(Hz)	Bytes Tx	Bytes Rx	P.Loss (%)
10	3906.25	16K	1048	93.45
50	781.25	16K	4509	71.82
100	390.62	16K	8318	48.01
250	156.25	16K	14176	11.40
500	78.13	16K	15796	1.28
700	52.08	16K	15939	0.38
750	39.06	16K	15882	0.74
1000	31.25	16K	15891	0.68
1250	19.53	16K	15905	0.59
2000	7.81	16K	15971	0.18
10000	3.91	16K	15966	0.21

Table 6.2: 802.11 Packet Loss

The results show that the Packet Loss is not very significative for frequencies lower than approximately 50 Hz. This is a very low frequency if one considers that the module has a theoretical maximum speed of about 2Mbps, however, it is a normal value given all the associated overhead and the fact that the involved packets only have 16-bytes of data, which is actually less than the overhead associated with each packet.

The figure 6.4 shows a part of the received packets log. Only the data part of the frame is shown and it shows the packets being correctly received.

Actually, not all of the packets were received without errors; some packets didn't arrive and the application couldn't read some of them properly. This has clearly a negative effect over the verified packet losses.

Even knowing that the results would be different in an environment with different characteristics or if the distance between the devices was different, these measurements serve well the purpose of indicating how the system behaves in an assumed agressive environment with a large number of wireless devices and, consequently, a great amount of collisions. A solution to increase the speed would be to reduce the percentage of overhead per 802.11 packet. This could be achieved by using UDP instead of TCP for the transfers or by sending a larger amount of data with each 802.11

packet. The UDP has a much lower overhead, but is also a less complex protocol (doesn't include, for example, acknowledgment mechanisms).

Chapter 7

Conclusions and Future Work

The present dissertation stands on the principle that low cost solutions may be used to address low resource-demanding situations, as is the case of localization of some unexpensive or non-critical objects or devices. For the localization of this kind of objects, there is no need to have a very reliable localization system if it implies a high cost or effort, both for the creation or purchasing and maintenance of the system. In particular, this dissertation is focused on the usage of the IEEE 802.15.4 standard and its integration with the IEEE 802.11 standard to assist, among other applications, in the localization of people or objects.

The dissertation has been divided into two main parts, being the first related to the study and conception of a system capable of performing RSSI measurements that could yield information about the distance between two low-cost and low-power (IEEE 802.15.4) wireless devices. The second part focused on integrating this low-cost and low-power technology with a higher-cost and more standardized technology (IEEE 802.11).

The main goals were attained, namely the creation of an 802.15.4-802.11 *Gateway*, which is the main core of the infrastructures that were created. The possibilities of application for such infrastructures are enormous, going from applications in the medical area and localization systems to domotic applications.

The negative points of these technologies (some of them common to many wireless technologies) are their dependence on external and out-of-control factors that may condition the used medium (air). There are, as well, some limitations imposed by the low power requirements of the 802.15.4 technology that, in turn, impose low transmission speeds and other limitations. Most of these limitations can, however, be ignored in some non-critical applications where there are not very hard temporal restrictions.

From the work developed throughout this dissertation and the various different tests performed, it became clear that the created infrastructures, even if not yet optimized for maximum performance, are able to serve the purpose of supporting localization systems, as long as they only need a reasonable, not exagerated, amount of resources. That is, for systems with a low to medium number of devices to be located and in moderatly agressive wireless environments, the communication between the various blocks of the system proved

itself stable and sufficient to handle the imposed work loads.

Obviously, there is still a long way to run, specially concerning the optimization of the system, to reach ideal performance. There are various situations that were not adressed (because they didn't fall in the scope of this dissertation) that may be attended in the future, in order to continue improving, not only the overall performance and speed of the system but also its energetic efficiency.

Some changes or improvements that could be done are:

- reducing the 802.11 overhead by using UDP instead of TCP;
- configuring the sleep mode in the lower-power devices, to achieve longer battery life;
- developing an adequate Medium Access Control protocol to avoid collisions among *Tag* transmissions in broadcast;
- further optimization of the software to allow, if necessary, for faster communication speeds
- implementation/activation of some security features to limit the access of hostile wireless devices to the created networks

Appendix A

Development Environment

To develop the necessary software for the correct functioning of the system, as the core hardware components are from *Microchip*, the obvious choice was a *Microchip IDE*, specifically the **Microchip MPLAB IDE**[7].

This software is free and provides an Integrated Development Environment with useful features when working with *Microchip* devices. It includes a C compiler (useful, as this was the chosen programming language) and even options for debugging and directly loading the software onto the target devices (when necessary, in this case a different loader was used though[8]).

To be able to compile code for the used platform (dspic33), the IDE must be provided with the C30 compiler (may be chosen during the first installation of the IDE). Having the IDE ready, it is intuitive to use, open projects and compile them, being that the Makefile for all projects is automatically created and updated by the IDE and all that the user has to do is include the correct headers in each source file.

The source code for both the *Gateway*s and textitTags is included, attached to this dissertation. The file organization differs between both projects, mainly due to the different characteristics of each one but also because both were based on some source code provided by the manufacturer (properly adapted, instead of having all the source code created from scratch).

Figure A.1 shows the file tree for each of the projects.

Figure A.1: Project files for *Tag*s (left) and *Gateway*s (right)

A.0.1 Opening and Compiling

Tag project

This project is based on the RadioDriver Application[9] provided by Microchip, with some modifications performed by *MicroI/O* to adapt the driver to the uMRF board. Some of the included files (like *Compiler.h*) are generic and necessary for the successfull compilation of the software. The most important files in this project (i.e. the ones where the board characteristics can be defined and operation preferences can be set) are:

- **MRF24J40.h** - defines the read and write registers of the MRF24J40 module by associating names with the respective memory addresses.
- **RadioDriverDefs.h** - specifies various definitions related to the MRF24J40 module and how it connects to the uMRF board. It is possible to define here the physical address of the module, the clock frequency, the microcontroller timer to be used by the module, the connections (Reset, Chip Select, Interrupt pin, ...) to the board, among other settings.
- **MSPI.h/MSPI.c** - include functions to control the SPI transfers between the microcontroller and the MRF24J40 module.
- **RadioDriver.c** - represents the core of the driver. Includes the main() function and most of the other necessary configuration, initialization and operation functions. This file includes the function to send packets (*broadcast*) where the packet frame format can be configured according to the needs of the application.

For each different *Tag*, the value of the BYTE type variable *Source_Short_Address* must be different as this is the field that identifies each individual *Tag*. (Note that this could impose a very low limit on the total number of *Tag*s and, for that reason, the BYTE type variable *Source_PAN_Identifier* was created. The combination by concatenation of the two referred variables greatly multiplies the number of possible *Tag*s. The *Source_PAN_Identifier* variable can be thought of as an extension to the *Source_Short_Address* variable and by changing its value, 256 more unique *Tag* IDs can be created. The name of the variable does not suggest it though, only because, as explained previously in the dissertation, all the fields were thought and created according to the ZigBee standard for compatibility reasons.)

Gateway project

This project is based on the TCPIP Stack Demo Application[10] provided by Microchip, adapted to work with the ZG2100 wireless module. Additionally, as the Gateway is supposed to include both the ZG2100 and the MRF24J40 modules, the radio driver used for

the *Tag*s was integrated into this project (for this reason, some files are the same or very similar to files in the *Tag* project and are not mentioned here).

- **HardwareProfile.h** - defines most of the uMRF board peripherals connections to the microcontroller (including the ZG2100 module). Performs a role similar to the RadioDriverDefs.h file (included in both projects).
- **Gateway.c** - core of the Gateway. Includes most of the initialization functions and others related to the transmission and reception of packets, as well as the necessary interrupt service routines.
- **TCPIP Stack Headers.h** - Configuration file for most things related to the 802.11 connection (through the ZG2100 module) and to the TCP/IP stack. Network addresses (IP), SSID, wireless security features and other settings can be configured in this file.

To open either one of the projects with the MPLAB IDE, the main project file (with extension .mcp) must be searched for in the project's root folder. The navigation through the menus and toolbars of the program is quite intuitive. All the attached code is ready to be opened and compiled (by pressing Ctrl+F10 or by using the toolbar buttons on the IDE), resulting in a file with extension .hex that may then be upladed to the uMRF board.

A.0.2 Uploading software to the uMRF board

After the project has been compiled and HEX file has been created that can be uploaded to the uMRF board with the assistance of another piece of software (ds30loader[8]). The board must be connected to the computer running the ds30loader by means of an USB cable. This creates a virtual serial port on the host computer (there is an USB to RS232 converter embedded in the board) which can then be used to transfer data/programs.

The ds30loader is a program that allows a host computer to communicate with a Microchip ds3x series PIC (assuming it has been flashed with a compatible bootloader) via a serial interface. The uMRF boards have been loaded with an appropriate bootloader which is able to connect via its serial port at a speed of **115200 bps**. The ds30loader software has plenty of options, being the most important ones the choice of the correct serial port, the correct speed (115200 bps, the same speed as the UART in the uMRF board) and the file to be uploaded. A general aspect of its main window is shown in figure A.2.

Figure A.2: General Aspect of the ds30loader application

After all the necessary parameters have been configured (serial port, speed and hex file to be downloaded by the board), the "“Download" button must be pressed, followed by the RST button (see fig. A.3) on the uMRF board (to reboot the board and enter

the bootloader). After the program has been successfully downloaded, it starts to run in the uMRF board and some status messages may be sent to the ds30loader application via serial port (useful to verify the correct initialization of the program).

Figure A.3: uMRF board (with ZG2100 module)

This procedure is the same for both the *Gateway*s and the *Tag*s. After at least one *Gateway* is loaded with the appropriate program, it becomes ready to listen any accepted *Tag*s that broadcasting messages and the system becomes ready to work.

Appendix B

Acronyms

AP Access Point

API Application Programming Interface

BSD Berkeley Software Distribution

CGI Common Gateway Interface

CRC Cyclic Redundancy Check

CSMA-CA Carrier Sense Multiple Access with Collision Avoidance

FCS Frame Check Sequence

FIFO First In, First Out

FSM Finite State Machine

GPS Global Positioning System

ICS Internet Control Suite

IDE Integrated Development Environment

IEEE Institute of Electrical and Electronics Engineers

IPC Inter Process Communication

ISM Industrial, Scientific and Medical (frequency band)

LAN Local Area Network

LQI Link Quality Indicator

MAC Medium Access Control

PDA Personal digital assistant

PHY Physical (Layer)

PIC Programmable Interrupt Controller

RAM Random-Access Memory

RF Radio Frequency

RSSI Received Signal Strength Indicator

SPI Serial Peripheral Interface

TCP/IP Transmission Control Protocol/Internet Protocol

UDP User Datagram Protocol

USB Universal Serial Bus

WLAN Wireless Local Area Network

WMN Wireless Mesh Networks

WSN Wireless Sensor Networks

Bibliography

[1] Microchip. *MRF24J40 Datasheet*, ds39776b edition, 2008.

[2] Patrick Kinney. Zigbee technology: Wireless control that simply works, 2003. http://www.zigbee.org/imwp/idms/popups/pop_download.asp?contentID=5162[cited07June2010].

[3] Kannan Srinivasan and Philip Levis. Rssi is under appreciated.

[4] Information Sciences Institute University of Southern California. Transmission control protocol - rfc793, 1981. http://tools.ietf.org/html/rfc793[cited07June2010].

[5] Carlos Eduardo Vieira Amador. Ap wi-fi/zigbee para suporte a localizacao baseada em redes sem fios, 2010.

[6] David Flowers and Yifeng Yang. Miwi wireless networking protocol stack - application note an1066. http://ww1.microchip.com/downloads/en/ AppNotes/MiWi%20Application%20Note_AN1066.pdf[cited07June2010].

[7] Microchip. Mplab ide. http://www.microchip.com/stellent/idcplg?IdcService =SS_GET_PAGE&nodeId=1406&dDocName=en019469&part=SW007002[cited07June2010].

[8] Mikael Gustafsson. ds30loader. http://sourceforge.net/projects/ds30loader/[cited07June2010].

[9] Microchip. Mrf24j40 radio utility driver program. http://www.microchip.com/stellent/idcplg?IdcService =SS_GET_PAGE&nodeId=1824&appnote=en535846[cited07June2010].

[10] Microchip. Tcpip stack. http://www.microchip.com/stellent/idcplg?IdcService =SS_GET_PAGE&nodeId=2680&dDocName=en537041[cited07June2010].